# Introducing Machine Learning

Agnieszka ŁAWRYNOWICZ [a] and Volker TRESP [b]
[a] *Institute of Computing Science, Poznan University of Technology, Poznań, Poland; E-mail: alawrynowicz@cs.put.poznan.pl*
[b] *Siemens AG, Corporate Technology, München, Germany; E-mail: volker.tresp@siemens.com*

**Abstract.** In this chapter we provide an overview on some of the main issues in machine learning. We discuss machine learning both from a formal and a statistical perspective. We describe some aspects of machine learning such as concept learning, support vector machines, and graphical models in more detail. We also present example machine learning applications to the Semantic Web.

**Keywords.** Machine Learning, Data Mining, Inductive Inference, Models

## Introduction

The goal of *Machine Learning (ML)* is to construct computer programs that can learn from data. The *inductive inference* of machine learning, i.e. the generalizations from a set of observed instances, can be contrasted to early Artificial Intelligence (AI) approaches that dealt mostly with deductive inference (cf. Krötzsch et al. [24] in this volume), i.e., the derivation of theorems from axioms. Although ML is considered a subfield of AI it also intersects with many other scientific disciplines such as statistics, cognitive science, and information theory[1]. An area, closely related to ML is *data mining* [11,12] which deals with the discovery of new and interesting patterns from large data sets. Although ML and data mining are often used interchangeably, one might state that ML is more focused on adaptive behavior and operational use, whereas data mining focusses on handling large amounts of data and the discovery of previously unknown patterns (implicit knowledge, regularities) in the data. Most of this chapter discusses ML in the context of a formal AI system, although when suitable, as in the discussion of graphical models, we assume a more statistical perspective.

ML approaches can be distinguished in terms of representation and adaptation. A machine learning system needs to store the learned information in some knowledge representation structure which is called (an *inductive*) *hypothesis* and is typically of the form of a *model*. Following the Ockham's razor principle, the hypothesis should generalize the training data giving preference for the simplest hypothesis; to obtain valid generalization, the hypothesis should be simpler than the data itself. A *learning algorithm* specifies how to update the learned hypothesis with new *experience* (i.e. *training data*) such that the *performance measure* with regard to the *task* is being optimized (see Figure 1).

[1]Certainly there are many different perspectives on machine learning: some researchers see the strongest link with statistics and even claim that both fields are identical.

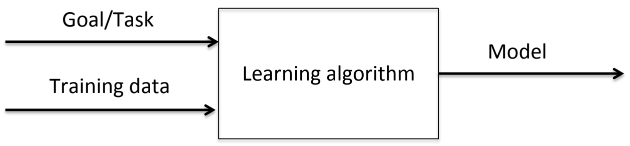


**Figure 1.** A generic machine learning method.

Over the years, machine learning methods have been applied to solve many real-world problems such as spoken language recognition, fraud detection, customer relationship management, gene function prediction etc. To provide a concrete example where machine learning has been effective on a Web service, consider the task of categorizing email messages as spam or non-spam, where the performance of the machine learning method is assessed by the percentage of email messages correctly classified. The training experience in this problem may come in the form of the database of emails that has been labeled as spam or no-spam by humans.

The rest of this chapter is organized as follows. In the next Section we discuss some basic aspects of machine learning and in Section 2 we discuss concept learning, the support vector machine and graphical models. In Section 3 we discuss applications of machine learning to the Semantic Web and Section 4 contains our conclusions.

## 1. Machine Learning Basics

In this section, we introduce the basic components of a machine learning problem. We discuss that learning algorithms are typically implemented as some kind of search and local optimization. The following Section 2 then describes three important machine learning approaches in more detail.

### *1.1. Tasks*

#### *1.1.1. Classification and Regression*

The tasks of *classification* and *regression* deal with the prediction of the value of one field (the target) based on the values of the other fields (*attributes* or *features*). If the target is discrete (e.g. nominal or ordinal) then the given task is called classification. If the target is continuous, the task is called regression. Classification or regression normally are supervised procedures: based on a previously correctly labeled set of training instances, the model learns to correctly label new unseen instances.

An example classification problem may consist in predicting whether to grant or not to grant a credit to a customer. The values of the class $c$ in this problem could be formed by a set {yes, no} representing a positive and a negative decision, respectively. The input to the classification method (that is, to a *classifier*) would consist of information about a customer. In particular, if the hypothesis space consists of rules, the output may be formed by a set of learned rules such as the one presented in Figure 2a.

### *1.1.2. Learning Associations*

An association describes a relation between objects, or measured quantities, that is the result of some interaction or of a dependency between the objects. Typically, the learned associations are in the form of *association rules* or sets of *frequent items*. The motivation for this type of task has been provided by market basket analysis where the methods for finding associations between products bought by customers are studied. For example, consider that customers who buy X (e.g. beer) typically also buy Y (e.g. chips); then, if we encounter a customer who buys X but does not buy Y, we may target this customer via cross-selling as a potential customer for Y. An itemset is called frequent if it appears in at least a given percentage (called *support*) of all transactions. Frequent itemsets are often the prerequisite for the learning of association rules.

### *1.1.3. Clustering*

*Clustering* is an unsupervised task, whose aim is to group a set of objects into classes of similar objects. A cluster is a collection of objects that are similar to each other within the same cluster, and dissimilar to the objects in other clusters. Therefore, an important notion in clustering (also known as *cluster analysis* in statistics) is the notion of *similarity* (or *distance*). In *conceptual clustering*, a symbolic representation of each cluster is extracted and we may consider each cluster to be a *concept*, closely related to a class in classification.

### *1.1.4. Other Machine Learning Tasks*

Some examples of other machine learning tasks are: reinforcement learning, learning to rank and structured prediction.

The reinforcement learning task consists of learning sequential control strategies. It deals with situations, where the output of the system is a sequence of actions that are performed to achieve some goal. An example may be game playing, where the complete sequence of moves is important, rather than a single move.

Learning to rank is a type of a (semi-)supervised learning problem where the goal is an automatic construction of a ranking model from training data, e.g., to learn to rank the importance of returned Web pages in a search application.

Structured prediction deals with prediction problems in which the output is a complex structure. Such problems arise in disciplines such as computational linguistics, e.g. in natural language parsing, speech, vision, and biology.

## *1.2. Training Data*

One distinguishes three important classes of feedback: feedback in the form of labeled examples, feedback in the form of unlabeled examples, or feedback in the form of reward and punishment, as in reinforcement learning.

*Supervised learning* consists of learning a function from training examples, based on their attributes (inputs) and labels (outputs). Each training example is a pair $(x, f(x))$, where $x$ is the input, and $f(x)$ is the output of the underlying unknown function. The aim of supervised learning is: given a set of examples of $f$, return a function $h$ that best approximates $f$. For example, given symptoms and corresponding diagnoses for patients, the goal is to learn a prediction model to make a diagnose based on symptoms for a new patient.

*Unsupervised learning* is concerned with learning patterns in the input, without any output values available for training. Continuing our example, only symptoms of patients may be available and the goal may be to discover groups of similar patients.

In *semi-supervised learning*, both labeled and unlabeled data is used for training, with typically only a small amount of labeled data, but a large amount of unlabeled data. In the clinical example, diagnoses might be available for only a few patients, and the goal would be to use this information for making most probable diagnoses for all patients.

In *reinforcement learning*, input/output pairs are not available to the learning system. Instead, the learning system receives some sort of a reward after each action, and the goal is to maximize the cumulative reward for the whole process. For example, in case of treatment planning, the learning system may receive reinforcement from the patient (e.g., feels better, feels worse, cured) as an effect of actions taken during a treatment.

Typically, a training data set comes in the simple form of *attribute-value* data table. However, more complex input data has also been studied, for example sequences, time series or graphs. From the point of view of this book, an interesting setting is presented by *Inductive Logic Programming (ILP)* [33,36], originally defined as a subfield of machine learning that assumes Logic Programming as a representation formalism of hypotheses and background knowledge. Since then ILP has been further refined to a broader definition that considers not only Logic Programs as a representation, but also other subsets of the first-order logic. In particular, its methodology is well-suited for the tasks where Description Logics are used as representation formalism. The distinguishing feature of the ILP methods is their ability to take into account *background knowledge*, that is the knowledge generally valid in some domain, represented, for example, in the form of ontologies.

## *1.3. Models*

Machine learning hypotheses may come in a variety of knowledge representation forms, such as equations, decision trees, rules, distances and partitions, probabilistic and graphical models. Classically, a division is made between *symbolic* and *sub-symbolic* forms of knowledge representation. The first category consists of representation systems in which the atomic building blocks are formal symbolic representations, often easily readable by a human. Such representation systems have compositional syntax and semantics, and their components may be assigned an interpretation. The system may, for example, be composed of a set of rules such as the one presented in Figure 2a. A good example of a symbolic system is an interpreted logical theory.

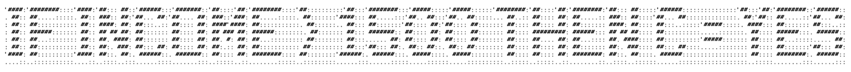
In turn, the components of a sub-symbolic representation system do not have a clear interpretation, and are not formal representations by themselves. Knowledge in this approach is represented as numerical patterns determining the computation of an output when being presented a given input. Good examples of sub-symbolic systems are neural networks, where the patterns are represented in the form of interconnected groups of simple artificial neurons.

Figure 2 provides an illustration of the distinction between symbolic and sub-symbolic representations.

Typical machine learning algorithms induce models, that is hypotheses that characterize globally an entire data set.

IF income $>$ 1500 THEN c = yes

(a)



(b)

**Figure 2.** An illustration of a) symbolic and b) sub-symbolic representation.

## *1.4. Generative and Discriminative Models*

So far the discussion was oriented towards a formal description of a learning problem. Here we describe a statistical view based on the concepts of generative and discriminative probabilistic models.

Generative models simulate a data generating process. In an unsupervised learning problem, this would involve a model for $P(X)$, i.e., a probabilistic model for generating the data.[2] In supervised learning, such as classification, one might assume that a class $Y \in \{0, 1\}$ is generated with some probability $P(Y)$ and the class-specific data is generated via the conditional probability $P(X|Y)$. Models are learned for both $P(Y)$ and $P(X|Y)$ and Bayes rule is employed to derive $P(Y|X)$, i.e., the class label probability for a new input $X$. In contrast, discriminative models model the conditional probability distribution $P(Y|X)$ directly, they learn a direct mapping from inputs $X$ to class label probabilities. To illustrate the difference between generative and discriminative models let us discuss an example task consisting in determining the language of a given speaker. In a generative modeling approach this task would be solved by learning language models for each language under consideration, i.e. by learning $P(X|Y)$ for each language $Y$ and by then applying Bayes rule to infer the language for a new text $x$. A discriminate model would not bother modeling the distribution of texts but would focus on the task of language classification directly and could focus on only the differences between languages. Popular examples of generative models are Naive Bayes, Bayesian Networks and Hidden Markov Models. Popular examples of discriminative probabilistic models are logistic regression and support vector machines.

## *1.5. Training Generative Models*

Since our description of graphical models is based on a generative modeling approach for an unsupervised model $P(X)$, we briefly discuss the training of such models. In a simple maximum likelihood model, one assumes a model $P(X|w)$, i.e. a probabilistic model for generating a data point given parameter vector $w$. The maximum likelihood parameters estimate is then defined by the parameters that maximize the likelihood, where the likelihood is $L(w)$ defined as the product of the probabilities of generating the $N$ independent training data points given the parameters and assumes the form

$$L(w) = \prod_{i=1}^{N} P(X_i = x_i|w)$$

[2]In the discussion on generative models, $X$ and $Y$ stand for random variables.

In a Bayesian approach the model is completed with an *a priori* distribution over models $P(M)$ and a prior distribution over parameters given model, i.e., $P(w|M)$. Based on observed data $D$, one can now calculate the most likely model as the one that maximizes

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

or the parameter distributions given model and data as

$$P(w|D, M) = \frac{L(w)P(w|M)}{P(D|M)}$$

A maximum a posteriori (MAP) estimate is achieved by taking the most likely model and selecting the parameters that maximize $P(w|D, M)$. A more truthfully Bayesian approach would consider the uncertainties in the estimates by integrating over unobserved quantities in the prediction.

## 2. An Overview of Selected ML Approaches

In this section, we provide an introduction to selected ML approaches, in particular to those that will be further referred to throughout the book.

### 2.1. Concept Learning

*Concept learning* consists of inducing the general definition of some concept (a category) given training examples labeled as members (positive examples) and nonmembers (negative examples) of the concept. Each training example consists of an instance $x \in X$ and its target concept value $f(x)$. Thus, a training example can be described by the ordered pair $(x, f(x))$. A learned concept is often represented as a boolean valued function. An example is called a positive one if $f(x) = 1$, and a negative one if $f(x) = 0$. Concept learning can be posed as a problem of searching the space of hypotheses to find a hypothesis best fitting the training data. The concept learning task may be thus formulated as follows [32]. Given instances $x \in X$, a target concept $f$ to be learned ($X \to \{0, 1\}$), hypotheses $H$, described by a set of constraints they impose on instances, training examples $D$ (positive, and negative examples of the target function), determine a hypothesis $h \in H$, such that $h = f(x)$ for all $x \in X$. Such a hypothesis, if learned on sufficiently large set of training examples, should also approximate the target concept well for new examples.

By choosing the hypothesis representation, one determines the space of all hypotheses that can ever be learned by the given method. The hypothesis space may be ordered by a generality relation $\succeq_g$. Let $h_i$ and $h_j$ be two hypotheses. Then $h_i$ is more general or equal to $h_j$ (written $h_i \succeq_g h_j$) if and only if any instance satisfying $h_j$, also satisfies $h_i$, where an instance $x \in X$ is said to *satisfy* $h \in H$ if and only if $h(x) = 1$. The relation $\succeq_g$ is a *partial order* (i.e., it is reflexive, antisymmetric and transitive) over the hypothesis space. Therefore, there may be also cases where two hypotheses are incomparable with $\succeq_g$, what happens if the sets of instances satisfied by the hypotheses are disjoint or intersect (are not subsumed by one another).

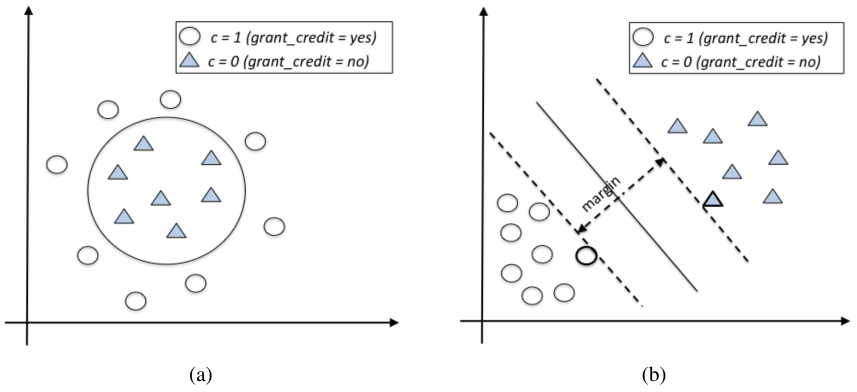


**Figure 3.** An illustration of SVMs. a) a non-linear, circular concept b) linearly separable instances, margin, and support vectors (with thicker border).

A hypothesis $h$ is called *consistent* with a set of training examples $D$ if it correctly classifies all these examples that is if and only if $h(x) = f(x)$ for each training example $(x, f(x))$ in $D$. The set of all hypotheses consistent with the training examples is called the *version space* $VS$ with respect to $H$ and the training examples $D$. The version space $VS$ may be represented by the sets of its maximally specific and maximally general hypotheses that delimit the entire set of hypotheses consistent with the data forming the boundaries of $VS$.

Concept learning algorithms utilize a structure imposed over the hypothesis space by the relation $\succeq_g$ to efficiently search for relevant hypotheses. For example, Find-S algorithm [32] performs the search from most specific to most general hypotheses in order to find the most specific hypothesis consistent with the training examples, while Candidate Elimination algorithm [32] exploits this general-to-specific ordering to compute the version space by an incremental computation of the sets of maximally specific and maximally general hypotheses. The search for hypotheses is steered also by the *inductive bias* of a concept learning algorithm, that is the set of assumptions representing the nature of the target function used by the algorithm to predict outputs given previously unseen inputs. The learning algorithm implicitly makes assumptions on the correct output for unseen examples to select one consistent hypothesis over another.

Some of concept learning algorithms proposed in the context of ILP that are relevant for this book are FOIL [39], and PROGOL [34]. They both induce first-order rules similar to Horn clauses. Concept learning is a very useful technique for ontology learning, and will be discussed in this context in more detail in (cf. Lehmann et al. [28] in this volume).

## 2.2. Support Vector Machines

*Support Vector Machines* (or *SVMs*) [2,3] may be used for binary classification or for regression. In binary classification, they construct a linear hyperplane (a *decision boundary*) to separate instances of one class from the other class. The separation between the classes is optimized by obtaining the separating hyperplane which is defined as the plane having the largest distance or margin to the nearest training data points of any class. Figure 3 illustrates the general concept of SVMs. Mathematically, the separating hyperplane may be defined by the equation:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

where $\mathbf{w}$ is a weight vector, $b$ is a scalar, and $\mathbf{w} \cdot \mathbf{x}$ is the inner product between $\mathbf{w}$ and $\mathbf{x}$.

The distance of the closest points to the decision boundaries *defines* the margin, which is maximized during training. The optimization problem solved by SVMs may be formulated as follows:

$$\begin{aligned}
&min_{\mathbf{w},\mathbf{b}} \mathbf{w} \cdot \mathbf{w} \\
&\text{where } \forall \mathbf{x} \in D : f(\mathbf{x})(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \\
&D \text{ is a set of examples, and } f(\mathbf{x}) = 1 \text{ for } c_i = 1, \text{ and } f(\mathbf{x}) = -1 \text{ for } c_i = 0
\end{aligned}$$

It can be shown that the margin is $\frac{1}{||\mathbf{w}||}$, where $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$. Thus minimizing $||\mathbf{w}||$, maximizes the margin. The optimization problem is usually not solved as posed in the Equation 1, but rewritten into a dual problem known as a constrained (convex) quadratic optimization problem, that can be solved by public domain quadratic programming solvers (for the details see for example [9]).

A new input $\mathbf{x}$ can be labeled as positive (1) or negative (0) based on whether it falls on or "above" the hyperplane:

$$\begin{aligned}
&\mathbf{w} \cdot \mathbf{x} + b \geq 0, \text{for } c_i = 1, \text{and} \\
&\mathbf{w} \cdot \mathbf{x} + b \leq 0, \text{for } c_i = 0
\end{aligned}$$

SVMs can also form nonlinear classification boundaries. For this purpose, the original input data is transformed into a higher dimensional space by a nonlinear mapping $\phi$, and a linear separating hyperplane in this new space is formed. Fortunately, the nonlinear mapping function $\phi$ does not need to be specified explicitly. While solving the quadratic optimization problem of the linear SVM, the training tuples occur only in the form of dot products, $\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$. Then, instead of computing the dot product on the transformed tuples, it is mathematically equivalent to apply a *kernel function* $K(\mathbf{x_i}, \mathbf{x_j})$ to the original input data, where the kernel is defined as:

$$K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$$

SVMs have attracted a lot of attention in recent years since they are less likely to suffer from overfitting than other methods. A drawback is that training scales unfavorable with the size of the training data set. This section mainly followed [11,40]. A good introduction to SVM methods may be also found in [3].

### *2.3. Learning in Graphical Models*

Given a set of $M$ features or random variables, $X_1, \ldots, X_M$, graphical models are a means to efficiently describe their joint probability distribution $P(X_1, \ldots, X_M)$ by exploiting independencies in $P(\cdot)$. We can consider graphical models as generative models, modeling the statistical dependencies among a potentially large number of variables. In terms of a knowledge representation, $X_i$ might stand for the truth value of a statement, and $X_i = 1$ if the statement is true and $X_i = 0$ otherwise.

In graphical models, independencies in the model can be displayed in form of a graph. We will consider the two most important subclasses of graphical models here, i.e., Bayesian networks (a.k.a directed graphical models) and Markov networks (a.k.a undirected graphical models).

### 2.3.1. Bayesian Networks

*The Basics* Any probability distribution of $M$ random variables can be decomposed in product form as

$$P(X_1, \ldots, X_M) = \prod_{i=1}^{M} P(X_i | \{X_j\}_{j<i})$$

where the order of the variables is arbitrary.

Bayesian networks exploit the fact that the set of all predecessors can be reduced to a set of parent nodes

$$\prod_{i=1}^{M} P(X_i | \{X_j\}_{j<i}) = \prod_{i=1}^{M} P(X_i | \text{par}(X_i))$$

where $\text{par}(X_i) \subseteq \{X_j\}_{j<i}$, thus exploiting independencies in a domain. In a graphical representation, nodes represent the random variables and one draws directed links from parent nodes to child node. Although the decomposition is possible in any order, most independencies are typically exploited when a causal ordering is observed, i.e, when the parents of a node also correspond to its direct causal probabilistic factors. It is of great importance to exploit domain independencies, since otherwise inference and other operations would require resources exponential in the number of variables. Consider, as an example, a diagnostic setting, where the random variables stand for diagnosis, symptoms and influencing factors. Influencing factors (e.g., smoking) are probabilistic causal factors for diseases, and diseases are probabilistic causal factors for symptoms. After the conditional probabilities $P(X_i | \text{par}(X_i))$ are defined by a medical expert, probabilistic inference can be used to calculate the probabilities of a disease given symptoms and given influencing factors.

One typically has the case that a certain probabilistic dependency appears several times in the training data. For example, this could describe the dependency of fever given flue, and this dependency is assumed identical for all patients. We use $P^k(X_i | \text{par}(X_i))$ where $i \in I(k)$ to indicate that the dependency between $X_i$ and its parents is of type $k$ and where $P^k(\cdot)$ now stands for a parameterized function.

Learning can assume varying complexity. In the simplest case, the causal structure and all variables are known in training and the log-likelihood can be written as

$$l(w) = \log L(w) = \sum_{k} \sum_{i \in I(k)} \log P^k(X_i | \text{par}(X_i), w)$$

where $w$ are model parameters. As we see, the log-likelihood nicely decomposes into sums which greatly simplifies the learning task. Typical learning approaches are maximum likelihood learning, penalized maximum likelihood learning, and fully Bayesian learning.

In the next level of complexity, some variables are unknown in the training data and some form of EM (expectation maximization) learning is typically applied.

Finally, we might also assume that the causal structure is unknown. In the most common approach one defines a cost function (e.g., Bayesian Information Criterion (BIC) or marginal likelihood) and one does heuristic search by removing and adding directed links to find a structure that is at least locally optimal. Alternatively one performs statistical testing to discover independencies in the domain and one defines Bayesian network structures consistent with those (constraint-based approach). Care must be taken, that no directed loops are introduced in the Bayesian network in modeling or structural learning. A large number of models used in machine learning can be considered special cases of Bayesian networks, e.g., Hidden Markov models, Kalman filters, which is the reason for the great importance of Bayesian networks. Readers, who are interested to learn more should consult the excellent tutorial [13].

*Modeling Relationships* Traditionally, Bayesian networks have mostly been applied to attribute-based representations. Recently, there has been increasing interest to applying Bayesian networks to domains with object-to-object relationships for which the term statistical relational learning (SRL) is used. Relationships add to complexity. In an attribute-based setting, one often assumes that objects are sampled independently, which greatly simplifies inference. For example the wealth of a person can be predicted from income and value of the person's home but, given this independence sampling assumption, is independent from the wealth of other people (given parameters). As a consequence, inference can be performed separately for each person. In SRL, one could also consider the wealth of this person's friends. As a consequence, random variables become globally dependent and inference often has to be performed globally as well, in this example potentially considering the wealth of all persons in the domain. A second issue is that directed loops become more problematic: I cannot easily model that my wealth depends on my friends's wealth and vice versa without introducing directed loops, which are forbidden in Bayesian networks. Finally, aggregation plays a more important role. For example, I might want to model that a given teacher is a good teacher, if the teacher's students get good grades in the classes the teacher teaches. This last quantity is probably not represented in the raw data as a random variable but needs to be calculated in a preprocessing step. As one might suspect, aggregation tends to make structural learning more complex.

Probabilistic Relational Models (PRMs) were one of the first published approaches for SRL with Bayesian networks and found great interest in the statistical machine learning community [23,10]. PRMs combine a frame-based logical representation with probabilistic semantics based on directed graphical models. Parameter learning in PRMs is likelihood based or based on empirical Bayesian learning. Structural learning typically uses a greedy search strategy, where one needs to guarantee that the ground Bayesian network does not contain directed loops.

Another important approach is presented by the infinite hidden relational model (IHRM) [50].[3] Here each object is represented by a discrete latent variable. The parents of a node representing a statement are the latent variables of all the objects involved. Thus, if $X_{u,m}$ stands for the statement that user $u$ likes movie $m$, then we would obtain the term $P(X_{u,m}|X_u^l, X_u^l)$ where $X_u^l$ is the latent variable for user $u$ and where $X_m^l$ is the latent variable for user $m$. The resulting network has by construction no loops. Also the need for aggregation is alleviated since information can propagate in the network of latent variables. Finally, no structural learning is required as the structure is given by the typed relations in the domain. The IHRM is applied in the context of a Dirichlet process mixture model where the number of states is automatically tuned in the sampling pro-

[3]Kemp et al. [20] presented an almost identical model independently.

cess. In [42] it was shown how ontological class information can be integrated into the IHRM and in [44] it is shown how OWL constraints can be integrated.

Bayesian networks are also being used in ILP where they form the basis for combinations of rule representations with SRL. A *Bayesian logic program* is defined as a set of Bayesian clauses [21]. A Bayesian clause specifies the conditional probability distribution of a random variable given its parents on a template level, i.e. in a node-class. A special feature is that, for a given random variable, *several* such conditional probability distributions might be given. For each clause there is one conditional probability distribution and for each Bayesian predicate (i.e., node-class) there is one combination rule. *Relational Bayesian networks* [17] are related to Bayesian logic programs and use probability formulae for specifying conditional probabilities.

### *2.3.2. Markov Networks*

*The Basics* The most important parameterization of the probability distribution of a Markov network is

$$P(X_1, \ldots, X_M) = \frac{1}{Z} \exp \sum_k w_k f_k(\{X\}_k)$$

where the feature functions $f_k$ can be any real-valued function, where $\{X\}_k \subseteq \{X_1, \ldots, X_M\}$ and where $w_k \in \mathbb{R}$. $Z$ normalizes the distribution. In a graphical representation, all variables in $\{X\}_k$ would be mutually connected by undirected links and thus would form cliques in the resulting graph.

We consider first that the feature functions $f_k(.)$ are given. Learning then consists of estimating the $w_k$. The log-likelihood is

$$l(w) = -\log Z + \sum_k w_k f_k(\{X\}_k)$$

Even a simple maximum likelihood estimate leads to a non-trivial optimization problem, since $Z$ is a function of all the parameters in the model.

The more complex question is, how application specific feature functions $f_k(.)$ can be defined. In Markov logic networks, as described next, the feature functions are derived from logical constraints.

*Markov Logic Networks (MLN)* Let us consider a simple example with two friends $A$ and $B$. Let $X_{A,r} = 1$ and $X_{B,r} = 1$ stand for the facts that person $A$ is rich, respectively person $B$. Let $X_{A,p} = 1$ and $X_{B,p} = 1$ stand for the facts that person $A$ is poor, respectively person $B$. Then we define the feature function $f_{r,r}(X_{A,r}, X_{A,r})$ which is only equal to one if $X_{A,r} = 1$ and $X_{B,r} = 1$ and is equal to zero else. Similarly, we define $f_{p,p}$, $f_{p,r}$, $f_{r,p}$. After training, we might obtain the weights $w_{r,r} = 10$, $w_{p,p} = 10$, $w_{r,p} = -5$, $w_{p,r} = -5$. Thus a situation where both friends are both rich or both are poor is much more likely (for example, $P(X_{A,r} = 1, X_{B,r} = 1, X_{A,p} = 0, X_{B,p} = 0) = Z^{-1} \exp 10$) than the situation where only one of the is rich and the other one is poor (for example, $P(X_{A,r} = 1, X_{B,r} = 0, X_{A,p} = 0, X_{B,p} = 1) = Z^{-1} \exp -5$). We can consider that the features were derived from the logical expressions, $X_{A,r} \wedge X_{B,r}$, $X_{A,p} \wedge X_{B,p}$, $X_{A,r} \wedge X_{B,p}$, $X_{A,p} \wedge X_{B,r}$. Obviously this knowledge-base is not even consistent, but for MLNs this would not hurt. After learning, only the true statements would survive with $w \rightarrow \infty$. Even better, statements which are often but not always true would obtain weights which reflect this frequency. This basic ideas is formalized in MLNs.

Let $F_k$ be a formula of first-order logic and let $w_k \in \mathbb{R}$ be a weight attached to each formula. Then a MLN $L$ is defined as a set of pairs $(F_k, w_k)$ [45] [7]. One introduces a binary node for each possible grounding of each predicate appearing in $L$, given a set of constants $c_1, \ldots, c_{|C|}$. The state of the node is equal to 1 if the ground atom/statement is true, and 0 otherwise (for an $Q$-ary predicate there are $|C|^Q$ such nodes). A grounding of a formula is an assignment of constants to the variables in the formula (considering formulas that are universally quantified). If a formula contains $Q$ variables, then there are $|C|^Q$ such assignments. The nodes in the Markov network $M_{L,C}$ are the grounded predicates. In addition the MLN contains one feature for each possible grounding of each formula $F_k$ in $L$. The value of this feature is 1 if the ground formula is true, and 0 otherwise. $w_k$ is the weight associated with $F_k$ in $L$. A Markov network $M_{L,C}$ is a grounded Markov logic network of $L$ with

$$P(\{X\} = \vec{x}) = \frac{1}{Z} \exp\left(\sum_k w_k n_k(\vec{x})\right)$$

where $n_k(\vec{x})$ is the number of formula groundings that are true for $F_k$. MLN makes the unique names assumption, the domain closure assumption and the known function assumption, but all these assumptions can be relaxed.

A MLN puts weights on formulas: the larger the weight, the higher is the confidence that a formula is true. When all weights are equal and become infinite, one strictly enforces the formulas and all worlds that agree with the formulas have the same probability.

The simplest form of inference concerns the prediction of the truth value of a grounded predicate given the truth values of other grounded predicates (conjunction of predicates) for which the authors present an efficient algorithm. In the first phase, the minimal subset of the ground Markov network is returned that is required to calculate the conditional probability. It is essential that this subset is small since in the worst case, inference could involve alle nodes. In the second phase Gibbs sampling in this reduced network is used.

Learning consists of estimating the $w_k$. In learning, MLN makes a closed-world assumption and employs a pseudo-likelihood cost function, which is the product of the probabilities of each node given its Markov blanket. Optimization is performed using a limited memory BFGS algorithm.

Finally, there is the issue of structural learning, which, in this context, defines the employed first order formulae. Some formulae are typically defined by a domain expert *a priori*. Additional formulae can be learned by directly optimizing the pseudo-likelihood cost function or by using ILP algorithms. For the latter, the authors use CLAUDIEN [41], which can learn arbitrary first-order clauses (not just Horn clauses, as many other ILP approaches).

## 3. Applications of ML to the Semantic Web

The interest of applying ML techniques in the Semantic Web context has been growing over recent years, and at the main Semantic Web conferences special ML tracks and workshops have been formed[4].

One of the applications of ML techniques that is discussed throughout this book is *ontology learning*. ML techniques may be used to both learn ontologies from scratch and enrich already existing ontologies. Learning data originates, e.g., from Linked Data, social networks, tags, textual data [30,8,27]. Another popular use of ML is the learning of the mapping from one ontology to another (e.g. based on association rules [5], or similarity-based methods [6] ).

A number of proposed approaches for *Learning from the Semantic Web* are based on ILP methods (e.g., classification [27] or association learning [19,26]). This kind of approach is supported by recently developed tools for ontology-based data mining such as DL-Learner [27][5], RMonto [38][6]) or SDM-Toolkit [49][7]. An interesting application of this kind was realized within the project e-LICO[8]. It consisted of optimizing knowledge discovery processes through ontology-based meta-learning, that is machine learning from meta data of executed past experiments, where meta data was represented with background ontologies [14][9]. [29] describes a perspective of ILP for the Semantic Web.

Ontology learning and ILP assume deterministic or close-to-deterministic dependencies. The increase of interest in ML techniques has arisen largely due to the open, distributed and inherently incomplete nature of the Semantic Web. Such a context makes it hard to apply purely deductive techniques, which traditionally have been dominating reasoning approaches for ontological data. As part of the LarKC project[10] a scalable machine learning approach has been developed that works well with the high-dimensional, sparse, and noisy data one encounters in those domains [47,15]. The approach is based on matrix factorization and has shown superior performance on a number of Semantic Web data sets [16]. Extensions have been developed that can take into account temporal effects and can model sequences [48] and can include ontological background and textual information [18]. The approach was part of the winning entry in the ISWC 2011 Semantic Web Challenge[11]. Tensor factorization is another promising direction, since the subject-predicate-object structure of the Semantic Web matches perfectly to the modes of a three-way tensor [35]. Another light-weighted approach is presented by [22], where the authors describe SPARQL-ML, a framework for adding data mining support to SPARQL. The approach uses relational bayes classifier (RBC) and relational probabilistic trees (RPT). In turn, [25] proposes to semantically group SPARQL query results via conceptual clustering.

An overview on early work on the application of ML to the Semantic Web can be found in [46] with applications described in [37]. Data mining perspectives for the

[4]Inductive Reasoning and Machine Learning for the Semantic Web (IRMLeS) workshops, `http://irmles.di.uniba.it`

[5]`http://aksw.org/Projects/DLLearner`

[6]`http://semantic.cs.put.poznan.pl/RMonto`

[7]`http://sourceforge.net/p/sdmtoolkit/`

[8]`http://www.e-lico.eu`

[9]`http://www.dmo-foundry.org`

[10]`http://www.larkc.eu`

[11]`http://challenge.semanticweb.org`

Semantic Web have been described by [1,31]. More recent overviews are presented in [4] and in [43].

## 4. Conclusions

In this chapter we have discussed machine learning as the basis for the remaining chapters in this book. With an increasing amount of data published in the format of the Semantic Web, we feel that the number of machine learning applications will certainly grow. Due to space limitations we could only cover a few aspects we felt *are* most relevant. By now machine learning is a large research areas with a multitude of theories, approaches and algorithms. We feel that there will not be one dominating approach towards machine learning on the Semantic Web but that we can expect creative solutions from different machine learning research areas.

## References

[1] Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards Semantic Web Mining. In *Proc. of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 264–278, London, UK, 2002. Springer-Verlag.

[2] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[3] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK, March 2000.

[4] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Inductive learning for the Semantic Web: What does it buy? *Semantic Web*, 1(1-2):53–59, 2010.

[5] Jérôme David, Fabrice Guillet, and Henri Briand. Matching directories and OWL ontologies with AROMA. In *Proc. of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 830–831, New York, NY, USA, 2006. ACM.

[6] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Ontology matching: A machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 385–404. Springer, 2004.

[7] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[8] Nicola Fanizzi, Claudia D'Amato, and Floriana Esposito. DL-FOIL concept learning in description logics. In *Proc. of the 18th International Conference on Inductive Logic Programming*, ILP '08, pages 107–121, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] Roger Fletcher. *Practical methods of optimization; (2nd ed.)*. Wiley-Interscience, New York, NY, USA, 1987.

[10] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pferrer, and Ben Taskar. Probabilistic relational models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[11] Jiawei Han and Micheline Kamber. *Data Mining. Concepts and Techniques*. Morgan Kaufmann, 2nd edition, 2006.

[12] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of data mining*. MIT Press, Cambridge, MA, USA, 2001.

[13] David Heckerman. A tutorial on learning with Bayesian Networks. Technical report, Microsoft Research, 1995.

[14] Melanie Hilario, Phong Nguyen, Huyen Do, Adam Woznica, and Alexandros Kalousis. Ontology-based meta-mining of knowledge discovery workflows. In Norbert Jankowski, Wlodzislaw Duch, and Krzysztof Grabczewski, editors, *Meta-Learning in Computational Intelligence*, pages 273–316. Springer, 2011.

[15] Yi Huang, Volker Tresp, Markus Bundschus, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the semantic web. In *Proc. of the 20th International Conference on Inductive Logic Programming*, ILP'10, pages 92–104, Berlin, Heidelberg, 2011. Springer-Verlag.

[16] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web Interoperability, Usability, Applicability (SWJ)*, 2012.

[17] Manfred Jaeger. Relational bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.

[18] Xueyan Jiang, Yi Huang, Maximilian Nickel, and Volker Tresp. Combining information extraction, deductive reasoning and machine learning for relation prediction. In *Proc. of the 9th International Conference on the Semantic Web: Research and Applications*, ESWC'12, pages 164–178, Berlin, Heidelberg, 2012. Springer-Verlag.

[19] Joanna Józefowska, Agnieszka Ławrynowicz, and Tomasz Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *TPLP*, 10(3):251–289, 2010.

[20] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Proc. of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 381–388. AAAI Press, 2006.

[21] Kristian Kersting and Luc De Raedt. Bayesian logic programs. Technical report, Albert-Ludwigs University at Freiburg, 2001.

[22] Christoph Kiefer, Abraham Bernstein, and André Locher. Adding data mining support to SPARQL via statistical relational learning methods. In *Proc. of the 5th European Semantic Web Conference*, ESWC'08, pages 478–492, Berlin, Heidelberg, 2008. Springer-Verlag.

[23] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *Proc. of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 580–587, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[24] Markus Krötzsch, Frantisek Simančík, and Ian Horrocks. A description logic primer. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[25] Agnieszka Ławrynowicz. Grouping results of queries to ontological knowledge bases by conceptual clustering. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, and Shyi-Ming Chen, editors, *ICCCI*, volume 5796 of *LNCS*, pages 504–515. Springer, 2009.

[26] Agnieszka Ławrynowicz and Jedrzej Potoniec. Fr-ONT: An algorithm for frequent concept mining with formal ontologies. In Marzena Kryszkiewicz, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Ras, editors, *ISMIS*, volume 6804 of *LNCS*, pages 428–437. Springer, 2011.

[27] Jens Lehmann. DL-learner: Learning concepts in description logics. *Journal of Machine Learning Research*, 10:2639–2642, 2009.

[28] Jens Lehmann, Nicola Fanizzi, and Claudia d'Amato. Concept learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[29] Francesca A. Lisi and Floriana Esposito. An ILP perspective on the Semantic Web. In *Semantic Web Applications and Perspectives, Proceedings of the 2nd Italian Semantic Web Workshop*, 2005.

[30] Alexander Maedche and Steffen Staab. Ontology learning. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 173–190. Springer, 2004.

[31] Peter Mika. *Social Networks and the Semantic Web*. Springer, 2007.

[32] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[33] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(3):295–318, 1991.

[34] Stephen Muggleton. Inverse entailment and PROGOL. *New Generation Computing*, 13:245–286, 1995.

[35] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proc. of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 809–816, New York, NY, USA, June 2011. ACM.

[36] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of inductive logic programming*, volume 1228 of LNAI. Springer, 1997.

[37] Hector Oscar Nigro, Sandra Gonzalez Cisaro, and Daniel Hugo Xodo. *Data Mining With Ontologies:*

*Implementations, Findings and Frameworks*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2007.

[38] Jedrzej Potoniec and Agnieszka Ławrynowicz. RMonto: Ontological extension to RapidMiner. In *Poster and Demo Session of the 10th International Semantic Web Conference*, Bonn, Germany, 2011.

[39] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[40] Luc De Raedt. *Logical and relational learning*. Cognitive Technologies. Springer, 2008.

[41] Luc De Raedt and Luc Dehaspe. Clausal discovery. *Machine Learning*, 26, 1997.

[42] Stefan Reckow and Volker Tresp. Integrating ontological prior knowledge into relational learning. Technical report, Siemens, 2007.

[43] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web - statistical learning for next generation knowledge bases. *Data Min. Knowl. Discov.*, 24(3):613–662, 2012.

[44] Achim Rettinger, Matthias Nickles, and Volker Tresp. Statistical relational learning with formal ontologies. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 286–301, Berlin, Heidelberg, 2009. Springer-Verlag.

[45] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[46] Gerd Stumme, Andreas Hotho, and Bettina Berendt. Semantic Web Mining: State of the art and future directions. *J. Web Sem.*, 4(2):124–143, 2006.

[47] Volker Tresp, Yi Huang, Markus Bundschus, and Achim Rettinger. Materializing and querying learned knowledge. In *Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web*, 2009.

[48] Volker Tresp, Yi Huang, Xueyan Jiang, and Achim Rettinger. Graphical models for relations - modeling relational context. In Joaquim Filipe and Ana L. N. Fred, editors, *KDIR*, pages 114–120. SciTePress, 2011.

[49] Anže Vavpetič and Nada Lavrač. Semantic subgroup discovery systems and workflows in the SDM-Toolkit. *The Computer Journal*, 2012.

[50] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel. Infinite hidden relational models. In *Uncertainty in Artificial Intelligence (UAI)*, 2006.